Università
della
Svizzera
italiana

Software Analytics - 2024

Prof. Gabriele Bavota
Alessandro Giagnorio

Automated Bug Triaging

Bug (or issue) triaging is an expensive and time consuming process consisting in assigning the "best-suited developer" to an open issue report. Widely used software projects can have hundreds of open issues per day. A striking example is the Microsoft vscode GitHub repository, counting more than 180k issues at date. The VScode project manager Erich Gamma[1] publicly stated that without tool support for the automated triaging of issues it would be impossible to handle the daily incoming flow of open issues. You can see the activities of the `VSCodeTriageBot` directly in GitHub.

Your **goal** is to build the best possible tool for bug triaging automation using machine/deep learning. The VScode GitHub repository will act as source to collect the data needed for training and testing your tool.

Technical Requirements

The tool is expected to take as input the id of an open VScode's issue on GitHub (*e.g.,* 228840) and provide as output a ranked list of candidate assignees extracted from the list of contributors of the corresponding repository (*i.e.,* microsoft/vscode). We consider a contributor a person who has been an assignee for past issues. The ranked list features the most likely assignee on top and, together with the candidate assignee name, should also feature the number of commits they authored in the VScode repository.

A shell-based user interface is sufficient, no fancy UI is needed (but you're welcome to create any UI you want).

As programming language you can use either Java or Python using any machine/deep learning framework of your choice. No other programming languages are admitted. Also, you can't mix in your project Java and Python.

Building the Training Set

As you know, to train a machine learner you need to build a training set. Your training set will be composed by all vscode's issues that (i) are closed; (ii) have exactly one assignee; and (iii) have an issue id ≤ 210000. The assignee of each issue represents the dependent variable to predict. In terms of predictor variables, you can exploit only what is available as soon as an issue is opened (*e.g.,* its title and description).

You are free to apply any text pre-processing you consider needed (*e.g.,* stopwords removal, stemming, synonyms matching, identifiers splitting, punctuation removal). Similarly, you may consider removing from the training set issues assigned to developers who have been assignees only a few times (this simplifies the learning of the model).

Training and Testing the Model

Once the training set has been built, you can build the test set using the closed VScode's issues with exactly one assignee and having an id going from 210001 to 220000.

We ask you to use the test set to evaluate the accuracy of two models. The first is trained on all instances of your training set. The second, only on "recent" instances, namely those having $190000 \leq id \leq 210000$ (note that this will also limit more the possible assignees the model can pick from).

[1] `https://en.wikipedia.org/wiki/Erich_Gamma`

A Few Notes

It's your job to come up with the solution that makes more sense to you. We are not going to grade your assignment based on the performance of the model you'll train, but we'll mainly look at (i) the care you put in cleaning and preparing the training set, (ii) the meaningfulness of the predictor variables you're going to use, (iii) the quality and clarity of the code you'll develop, and (iv) the usability (and proper functioning) of the tool you will deliver.

Submission Instructions

🗓 **Deadline:** Friday, October 25th, 2024 @ 19:00.

Each group must submit on iCorsi:

1. A textual document describing the data collection and cleaning process that has been performed, the attempts made in terms of predictor variables and the reasoning behind them. The document must also report the results achieved with the two different training strategies described above (only recent issues *vs* all issues). Additional analyses looking into the results are welcome (*e.g.*, Is it possible to improve its usability by filtering only the recommendations having a high confidence?).

2. A README explaining how to run the tool.